Beyond notebooks:
The other side of Jupyter

C.A.M. Gerlach
Hendrik D. Louzada
Carlos Cordoba
JupyterCon 2025

C.A.M. Gerlach

- Ex-atmospheric science researcher
- Full-time FOSS in Huntsville, AL 🚀
- Spyder core dev since 2017
  (Docs, UX, grants, DevOps, admin…)
- CPython core dev since 2023
  (PEP editor, Docs team, packaging…)






Hendrik Louzada

- Spyder core developer since 2023
- Primary developer of Spyder's remote development architecture
- Physics researcher at University of Sao Paulo, BR 🇧🇷
- Maintain PyMR library for medical imaging research



Carlos Cordoba

- Spyder core dev since 2010; project leader since 2013
- Built UI for remote development
- Now full-time FOSS in Bogota, CO 🇨🇴
- Ph.D in Industrial Engineering



Spyder:
A desktop frontend for Jupyter

- Python scientific environment and IDE
- Started in 2009 by Pierre Raybaut
- Developed by and for an international community of scientists & engineers
- Works on Windows/Linux/macOS
- Ecosystem of dependencies & plugins



Outline

1. QtConsole: A Jupyter frontend for desktop apps
2. Frontend/kernel interactions with Jupyter comms
3. Remote development with Jupyter server





1.

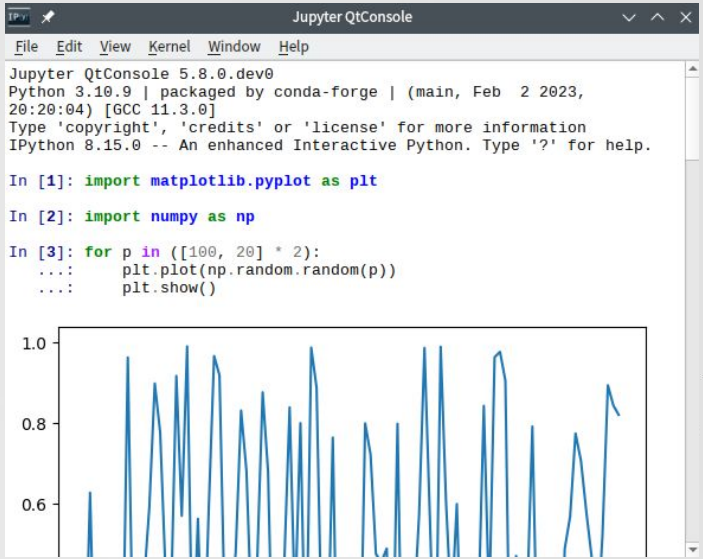
QtConsole: A Jupyter frontend for desktop apps

What is QtConsole?

- The first Jupyter frontend! (2011)
- Uses the Qt framework
- Used by many desktop scientific apps:
  Spyder, Napari, ChimeraX, NeXpy
- Terminal-like features
- Maintained by the Spyder team since 2017

QtConsole Features

- Add interactive console to Qt apps
- Supports all Jupyter languages
- Very similar to IPython
- Inline support for Matplotlib (static PNGs)
- In-process kernel to interact programmatically with the app



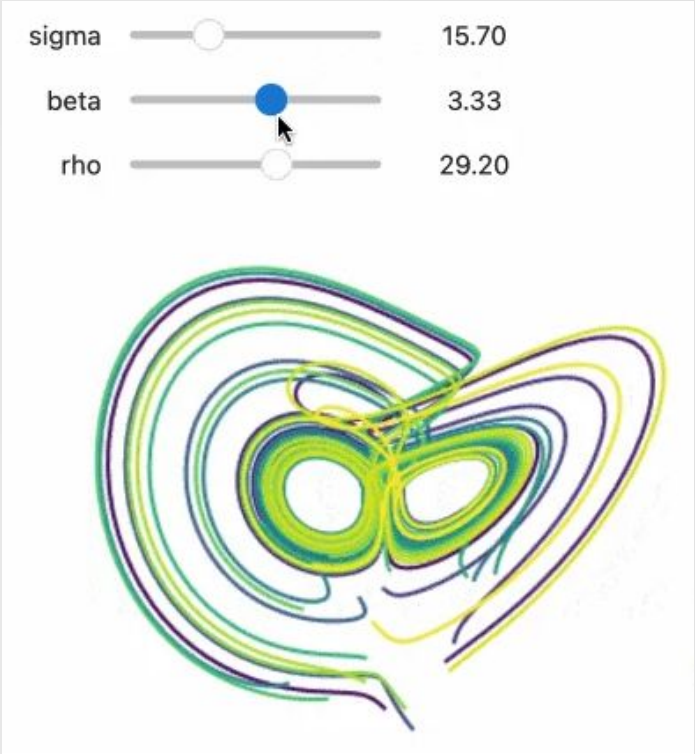


2.

Frontend and kernel interactions with Jupyter comms

What are comms?

- Protocol for two-way messages sent between Jupyter frontends and kernels
- They allow for real-time communications between those separate processes
- Used by many scientific apps or frameworks built to run in notebooks: IPywidgets, Panel, HoloViews, Bokeh

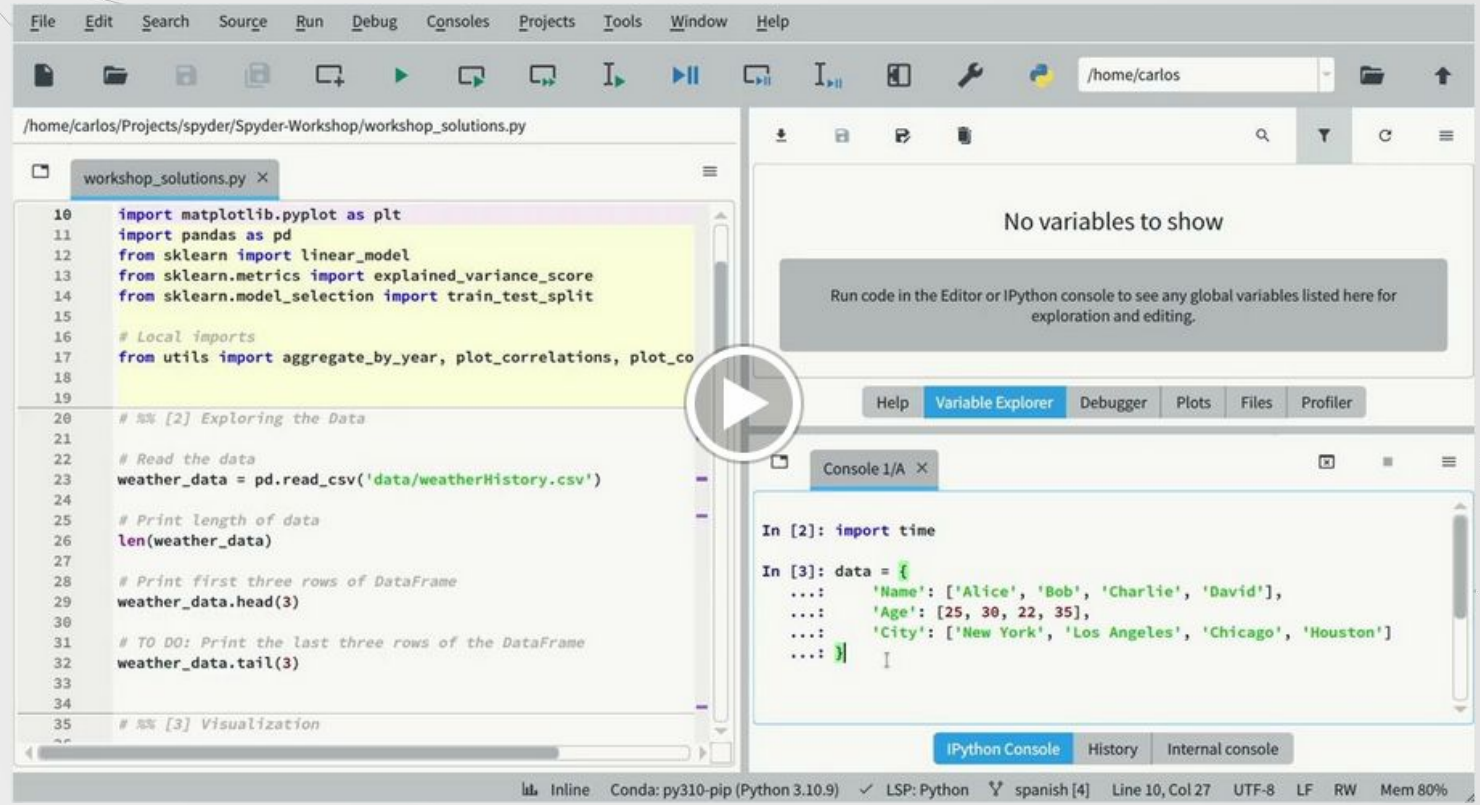


Advanced interactions with comms

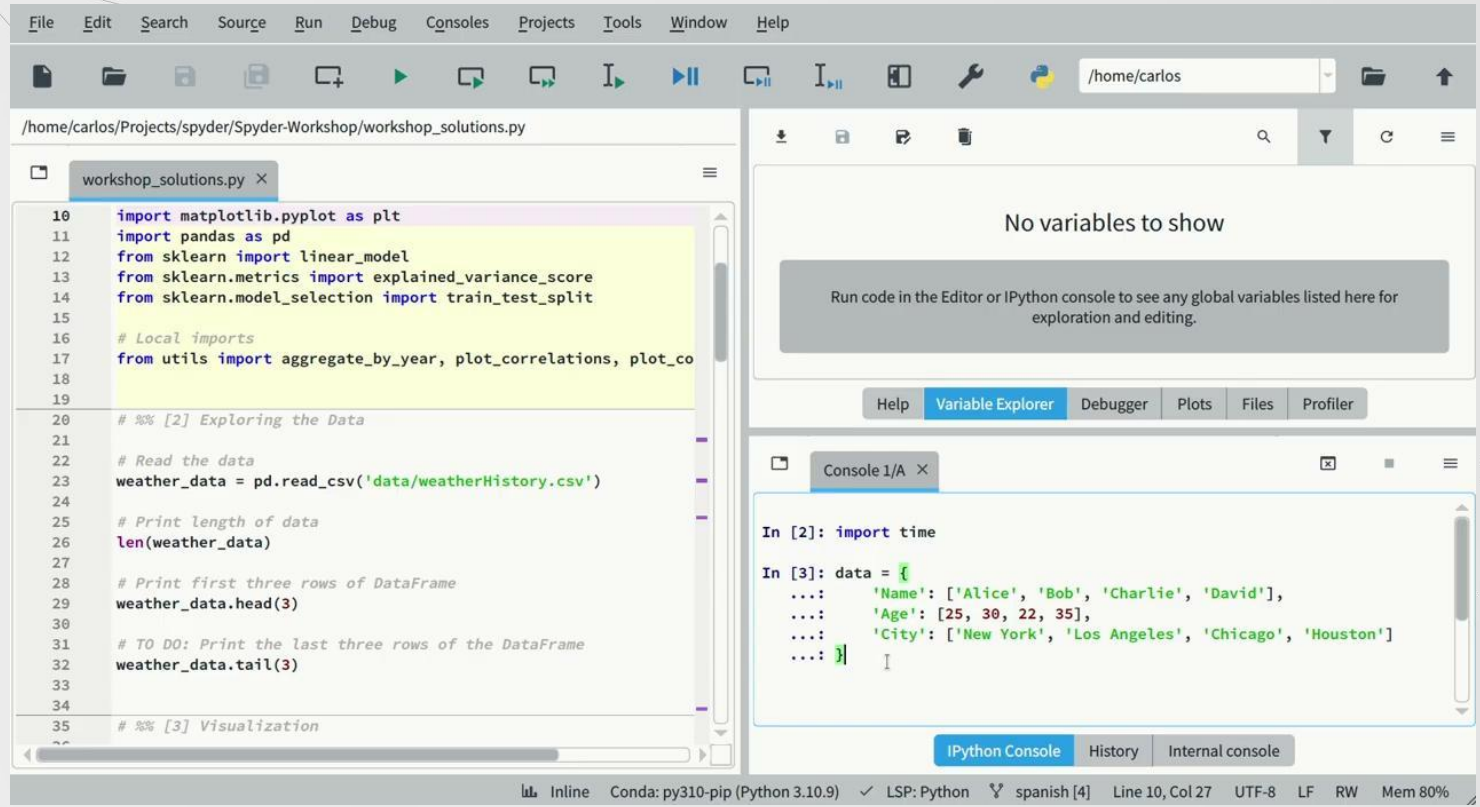
- Spyder has an architecture on top of comms so frontend can communicate with the kernel and vice versa
- These calls can be blocking, time out and pass the results to another callback
- On the frontend side, a call can also interrupt the kernel while running



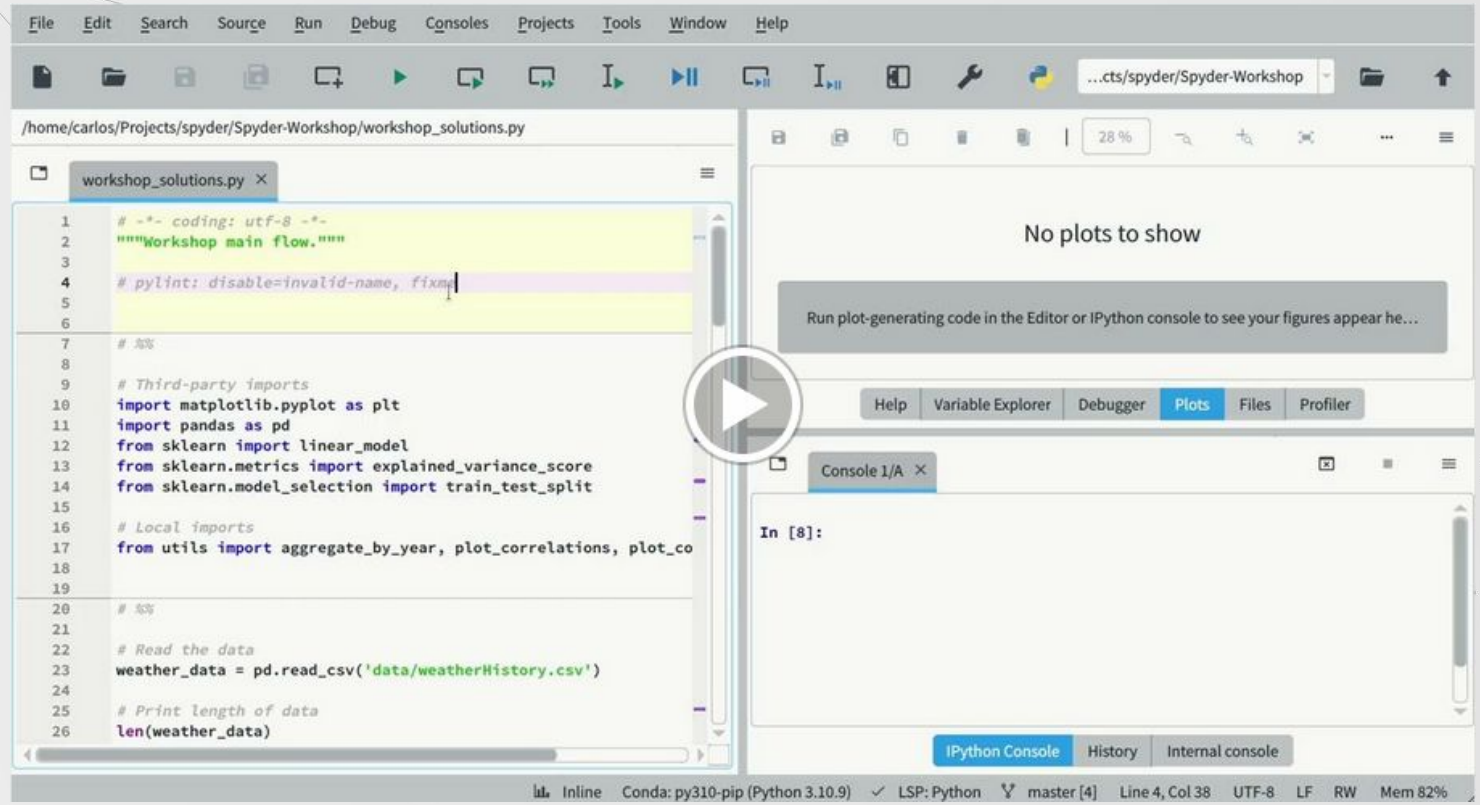
The Variable Explorer



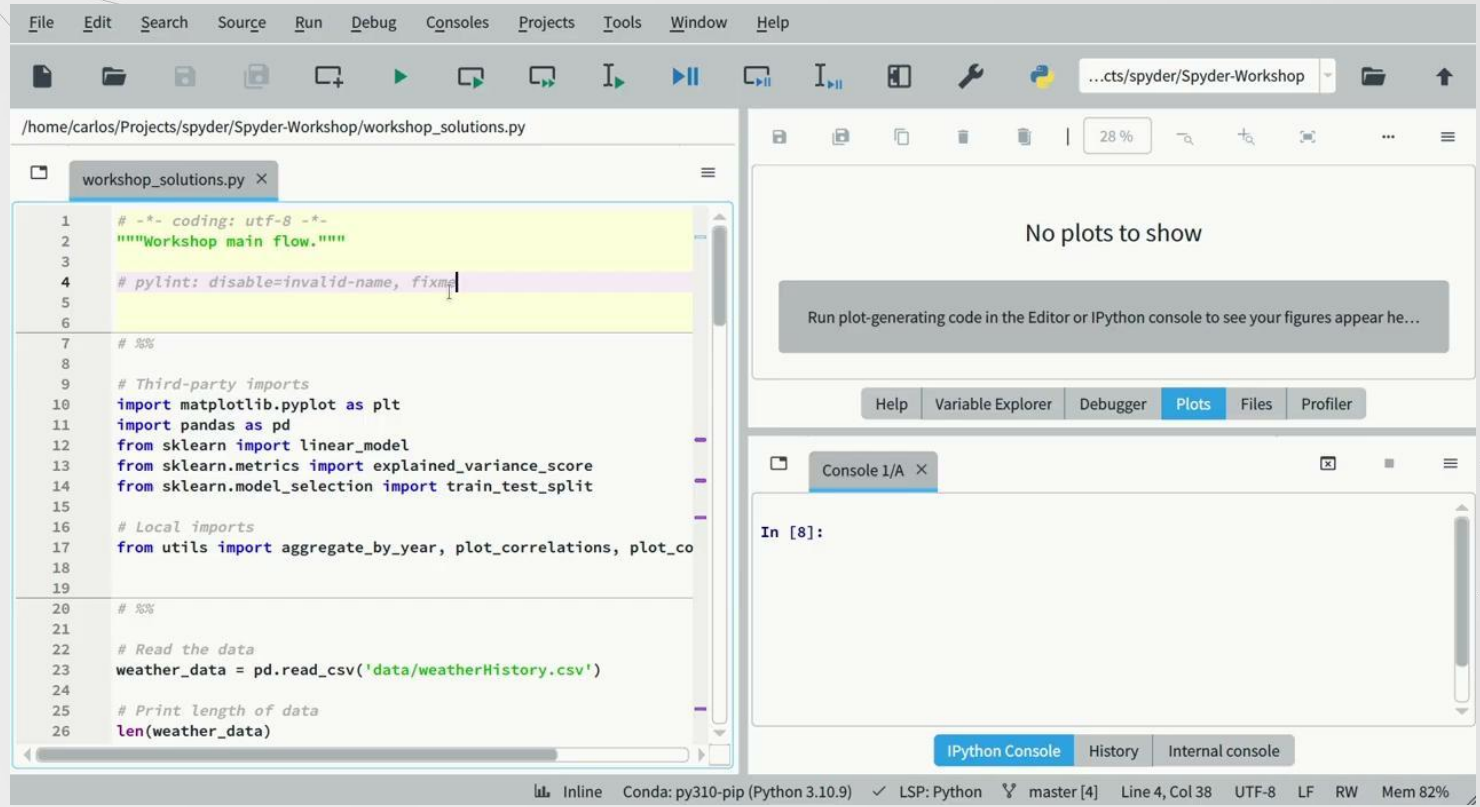
The Variable Explorer



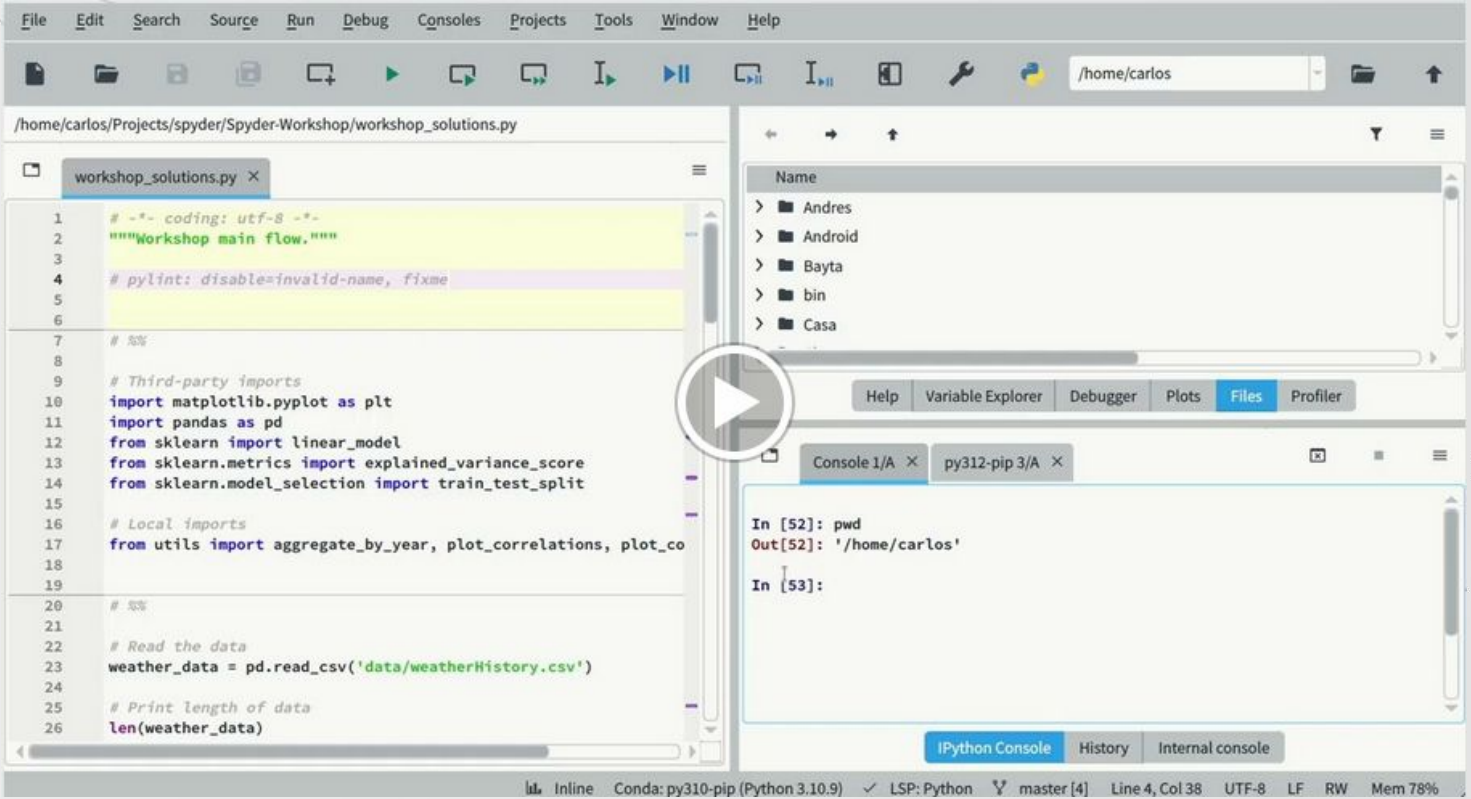
File/cell execution



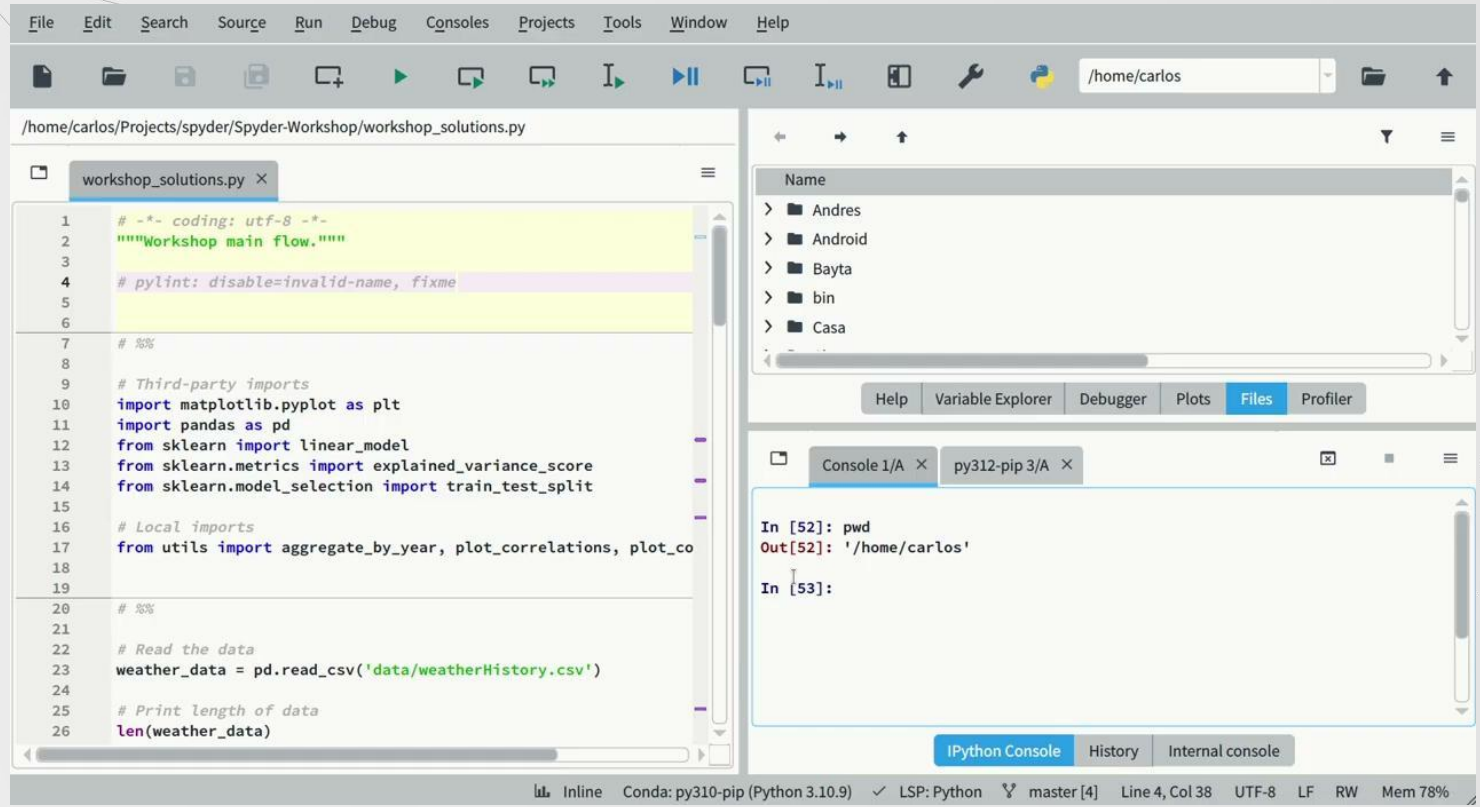
File/cell execution



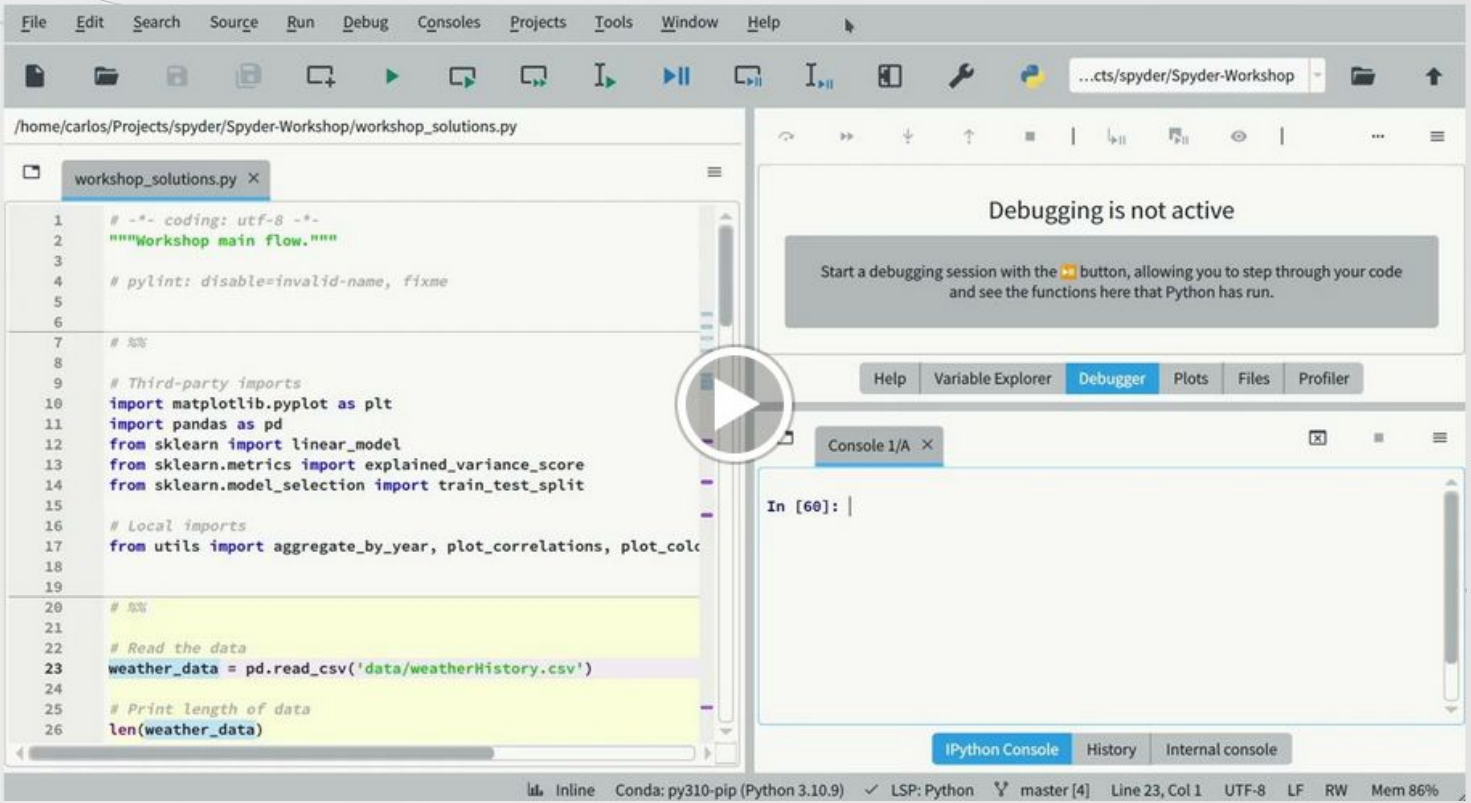
Display kernel info



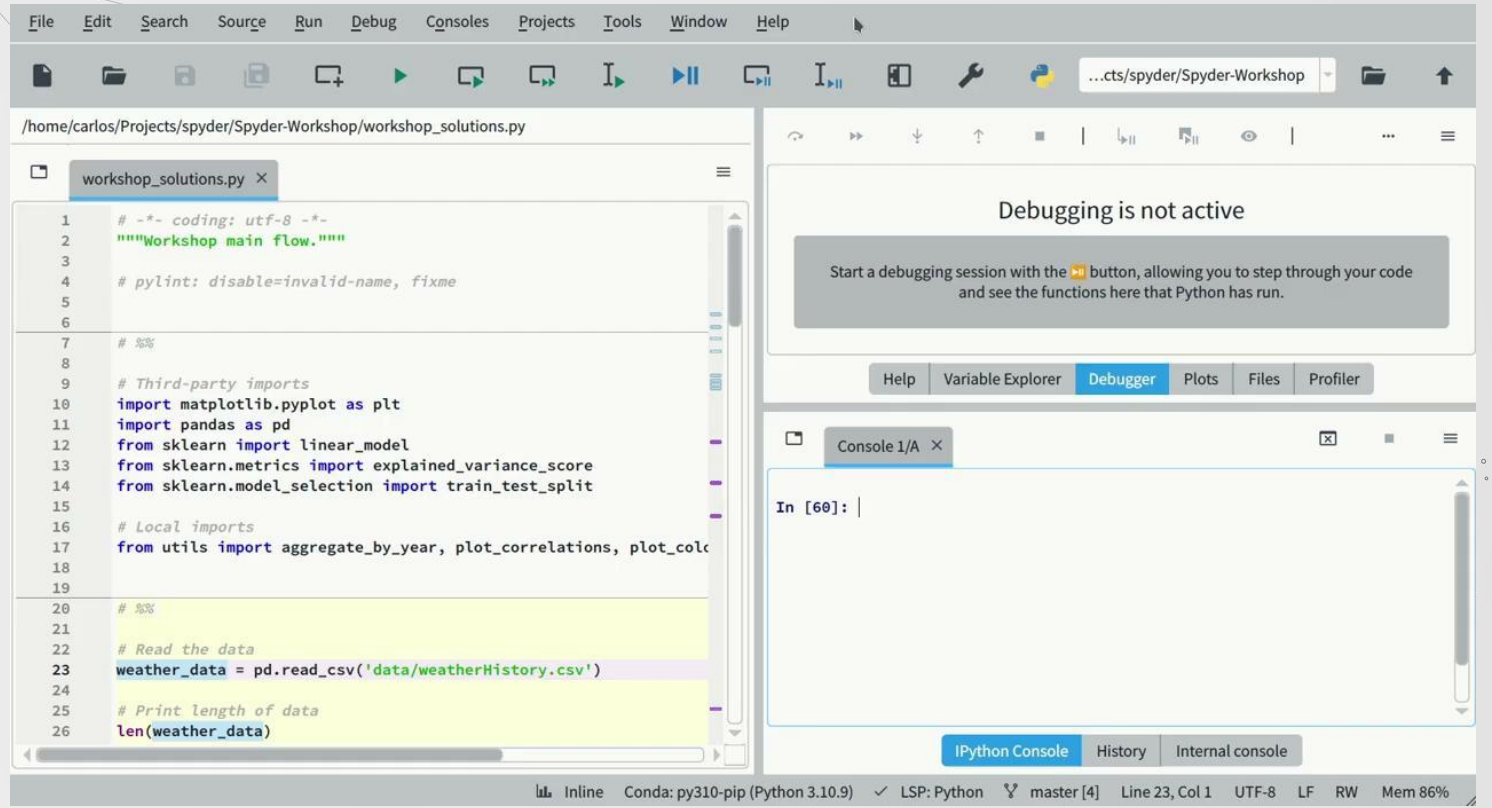
Display kernel info



The Debugger



The Debugger



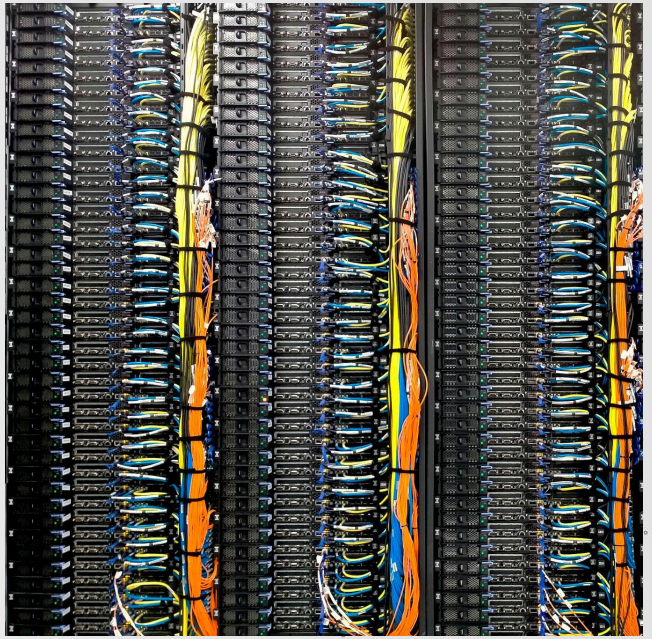


3.

Remote development with Jupyter Server and JupyterHub

The promise of the cloud

- Cloud and HPC clusters are **critical resources** for data-intensive research
- **Massively accelerates** computation
- Allows working with very **large datasets**
- Can dramatically **shorten the feedback loop** for data science and ML research
- **How to leverage** these capabilities?



Amfeli via Wikimedia Commons, cropped – CC-BY 4.0

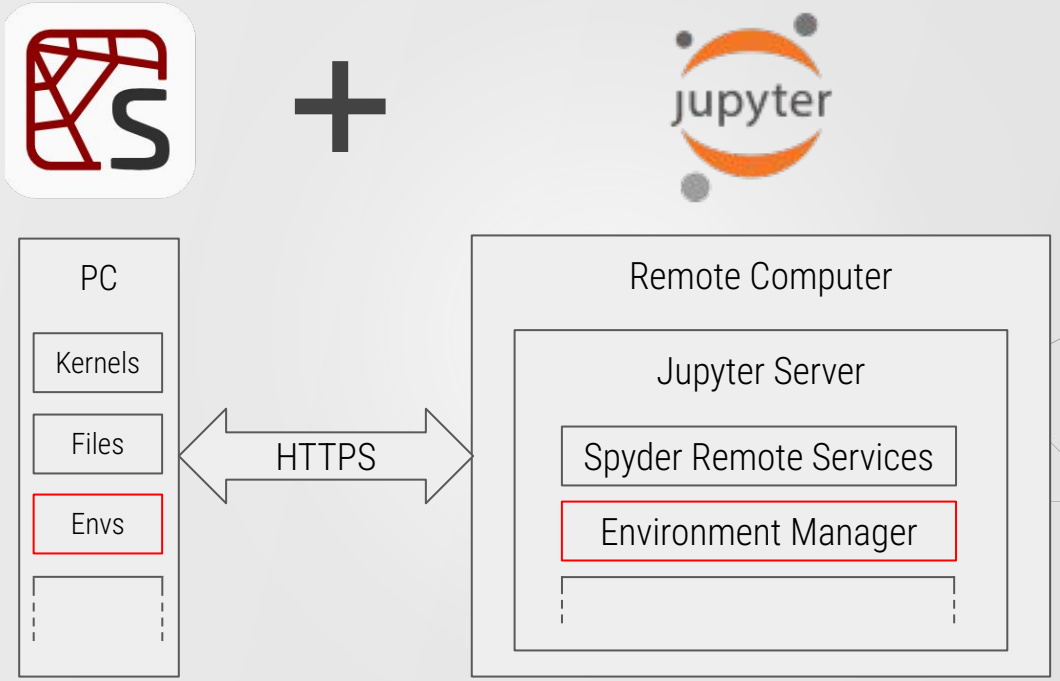
JupyterLab with JupyterHub

- Friendlier UI/UX for end users
- Locked to notebooks only
- Limited software development tools
- For most people **requires IT setup**, but IT is either **small or non-existent** for many users, esp. the **Global South**



jupyterlab

Spyder Remote Services



Spyder Remote Client

SSH Integration

Auto-installs and launches the remote server if missing

Establishes a secure tunnel to the remote machine

JupyterHub

Connects directly to use the Jupyter server it manages

Modular API System

Kernel Management
Remote file operations
} Spyder Remote Services

Extensible architecture to build custom REST API modules



Features

- Remote **IPython consoles**
- Remote **file system** explorer
- Extend with **plugins**
- Connect to **JupyterHub** servers
- Remote **Python environments** (released in 6.1.3)
- Remote **editing, terminals, projects** (#TODO - 6.2)



From zero to remote development in five minutes flat

1. Enter SSH credentials
2. Decide what packages to install
3. Wait for Spyder Remote Services to be installed and env created
4. Run your code!



Demo

Acknowledgements

Remote development work funded by **CZI EOSS 5** grant #384

Thanks to Jitse Niesen, Ryan Clary, Daniel Althviz and
the whole Spyder community for making this possible!



Happy Spydering!



spyder-ide.org